ChatGPT

Epic Integration Multi-Tenancy: Client IDs and Secrets Per Organization

Per-Organization Epic Credentials and App Registration

Epic's integration model is inherently **multi-tenant**, meaning each healthcare organization (hospital or practice) you connect must be handled as a separate "tenant" with its own Epic environment and API credentials. In production use, **every Epic customer maintains its own FHIR endpoints, app registration, and security scopes** – there isn't a single universal credential that grants access to all sites [1]. Instead, your app's Epic **Client ID** has to be enabled in each customer's Epic system individually. Epic's documentation confirms that *"the client ID must then be distributed to each customer environment the app connects to."* [2] In practice, this means:

- **Unique Endpoints:** Each hospital's Epic instance has a unique base FHIR URL (and authorization endpoints) that your app must point to.
- **Isolated Credentials:** Each organization needs to have your app's client record present in their Epic environment. The **Epic Client ID and Client Secret are effectively unique per customer site**. (Typically, you register your app once on Epic's developer portal to get a Client ID, then each hospital "downloads" that ID into their system. If an organization requires a separate configuration or subset of APIs, you might even create a distinct Client ID for that context, but generally the same ID can be reused with each customer's environment.)
- **One-by-One Onboarding:** "Going live" is not a one-click global switch – **you onboard each Epic customer separately**. As one guide notes, *in production, every Epic customer you integrate with could host its own endpoint URLs, its own app registration, and its own security scopes* [1]. Your rollout will be a sequence of customer-specific go-lives, each with its own configuration and testing.

In practical terms, when a new healthcare provider wants to use your Health Prep app, that organization (the Epic **community member**) must **authorize and install your app's Client ID into their Epic instance**. Epic provides a **Showroom/App Market** interface for this: for example, *each Epic customer that you contract with would download the Client ID(s) for your app through Epic's web interface (currently called "Showroom")* [3]. Once they do this, their Epic system knows about your application and will accept OAuth requests using the Client ID (and a matching secret). This process ensures **data segregation** – your app will only be able to query patient data from that specific organization's environment using the credentials they've enabled.

Sandbox vs. Production Credentials

It sounds like you're currently testing with Epic's **sandbox** credentials (the non-production Client ID/Secret provided for the Epic development sandbox). This is normal for development: Epic's sandbox is a shared testing environment (often a global endpoint like `fhir.epic.com` with synthetic patients). In the sandbox

phase, you use the **Epic-provided test client ID and secret** to simulate a single organization. However, **real-world production use requires separate credentials per site**. When you move to production:

- You will obtain your **production Client ID (and secret)** from Epic's App Orchard/Vendor Services after registering your app (if not already). This production ID will be used with real Epic customer systems.
- **Each customer organization must enable that Client ID in their environment.** In some cases (if your app meets certain criteria), Epic can **auto-distribute** your client ID to all customers for easy enablement [4]. Otherwise, the hospital's IT team will manually add or request your app. Either way, your **Client ID becomes known to their system**, and a Client Secret (for confidential apps) is set up for that site.
- The **Client Secret may be unique per site** depending on how distribution is handled. Often, as the developer you provide a client secret via the Epic portal that is shared, but some organizations can request a custom secret or configuration. The key point is that each instance will have a record of your app and an **authentication trust established via Client ID/Secret for that environment**.

As a result, your assumption that **each practice gets its own Client ID and Secret (scoped to their patient population)** is essentially correct. In fact, it's a necessary approach given Epic's architecture. A shared sandbox can simulate multiple orgs in development, but **in production each hospital or practice's Epic is a silo** – you connect to each silo with the appropriate credentials [5]. Put differently, **multi-tenancy in Epic integration is achieved by configuring your app separately for each client organization's Epic system**.

Multi-Tenant Architecture in Health Prep

From a software design perspective, you've already prepared for multi-tenancy. Your Health Prep application maintains an **Organization model with Epic credentials per organization**, which is exactly what's needed. In your `.env` template and code, it's noted that *Epic OAuth Client Credentials are managed in the Organization model for multi-tenancy* [6]. This means you will store each client's Epic **FHIR base URL**, **Client ID**, and **Client Secret** in your database (likely encrypted) and use those for API calls to that client's Epic system. For example, one hospital's config might have:

- `epic_fhir_url` = *https://<hospital>.epic.com/FHIR/api/...* (their unique FHIR endpoint)
- `epic_client_id` = *ABC-123-...* (the ID you gave them or they obtained for your app)
- `epic_client_secret` = *XYZ...* (the secret they received for that app record)

When an authorized user at that hospital uses Health Prep, the app will initialize the FHIR client with that hospital's settings. Another organization's users would trigger the app to use a different base URL and credentials. This **segregates data and authentication per organization** – exactly aligning with Epic's security model. Your code confirms no global default credentials are used in production; it forces an organization-specific config to be present [7] [8]. (The only default is for sandbox/testing via environment vars, which is fine for development.)

What about multiple departments within one hospital? If a single hospital has several departments using the app, typically they *share one Epic environment and one set of credentials*. Epic client IDs are usually issued per *instance* (often one per hospital or health system). You wouldn't normally have separate Client IDs for, say, Cardiology vs. Pediatrics **if they are part of the same Epic system**, since the data access can

be controlled by user roles and the API's patient context. All those departments' schedules reside in the same Epic database, so one integration (Client ID) can serve the whole hospital's patient population. In that case, you would onboard the hospital once, and all departments under it would use the same integration. Your app can still **"compartmentalize" by filtering data** (for example, only pulling appointments for the specific department or provider in context), but technically the *credentials* don't need to be different. However, if by "multiple departments" you mean distinct practices with separate Epic instances or separate administrative units that prefer isolation, it's possible to register multiple app credentials. Epic allows creating multiple Client IDs for different contexts of an app (e.g. one for a clinic affiliate vs. one for the main hospital, or separate "contexts" like a provider-facing vs patient-facing component) [9] [10]. In summary, **one Epic environment = one Client ID (generally)**. If a hospital's departments all use one Epic system, they will use one client ID/secret. If they had totally separate Epic servers, then indeed they'd use separate IDs and secrets as if they were separate customers. Your multi-tenant design can accommodate both scenarios (you would either create one Organization entry for the whole hospital or separate ones if truly needed).

Onboarding Workflow for New Organizations

To ensure your understanding is complete, here's how the onboarding typically works step by step (much of which you're already doing):

1. **Register Your App with Epic** – As a vendor, you sign up on Epic's developer site (App Orchard or the FHIR portal) and register your application. You'll get a **Non-Production (Sandbox) Client ID** (for testing) and a **Production Client ID** once approved. You also define redirect URIs, scopes, etc., during registration. This step is often accompanied by a review and a **Data Use Questionnaire** for what data your app will access [11].
2. **Obtain Client Credentials** – For confidential apps, you'll have a client secret or certificate. Epic's system provides a way to set a client secret for your app. For example, in your case you have an Epic sandbox client secret already for development. For production, you'll get a secret tied to the production Client ID. *(Patient-facing apps might not use a secret if they are public/OAuth implicit, but since Health Prep is provider-facing, it's a confidential client using OAuth code flow, hence a secret.)*
3. **Distribute Credentials to Customers** – When you sign a new hospital or practice, that organization must get your app's Client ID into their Epic. As noted, if your app meets certain criteria, **Epic can automatically list it for all customers** (so-called *auto-download*) [12]. Otherwise, you or the client will provide the Client ID to their Epic administrator. The admin will use Epic's **App ** request or "Purchase" process to add your app. (They typically search by the Client ID or app name in Epic's App market UI [13].) Upon adding it, the hospital's system will register the app and either generate or expect the** Client Secret**. Often, the developer uploads the secret in Epic's portal so that when the client "downloads" the app, the secret is delivered securely [14]. In other words, the** prospective organization works with Epic to obtain the Client ID and secret** for your app in their environment – exactly as you're guiding them to do on your site (contacting their Epic representative) [15]. This typically takes a few business days for the paperwork and setup.
4. **Configure the Organization in Your App** – Once the client has their Epic credentials, you (or they, via an admin UI) enter those values into your Health Prep system. This includes the Epic FHIR base URL for that site (each Epic customer will give you the proper FHIR endpoint URL for their system's version or you discover it via their SMART OAuth metadata endpoint), the Client ID, and Client Secret. Your application encrypts and stores these in the Organization record [16] [6].

5. **Testing and Go-Live** – With the credentials in place, you perform a connection test (your app's Epic integration status page can attempt an OAuth flow or a basic patient query to ensure things are working). Each site may have different data or slight config differences, so you verify that your queries (for schedules, etc.) return the expected results from that organization's Epic. After a successful test, that organization is "live" in your multi-tenant system, and Health Prep will pull in their patient scheduling data and other info via the FHIR API, using the OAuth tokens obtained with that site's credentials.

Throughout this process, **your design of separate client IDs/secrets per practice is confirmed as the correct approach**. Epic's own guidelines reinforce that each client app instance is unique to a customer: *"client records and client IDs play an important role in enabling Epic Community Members to control which applications access their Epic systems"* [17]. By partitioning credentials and connections by organization, you ensure that one client's data cannot be accessed by another's credentials, and you align with Epic's requirement that **each healthcare provider remains in control of their own data sharing**.

Conclusion

Yes – **your understanding is accurate**: multi-tenancy in the context of an Epic-integrated app means **compartmentalizing each customer's connection with its own Client ID/Secret and endpoint configuration**. A hospital or practice will supply you with credentials specific to their Epic instance, and your software uses those to access only that population's data. You will repeat this for every organization that onboards. In your implementation of Health Prep, the multi-tenant structure (storing credentials per organization) is designed to handle this reality [6]. Each new client organization will get their own Epic API credentials from Epic (with your guidance) and plug them into the app, effectively **creating a silo for that organization's patient schedules and data**. This approach not only **"compartmentalizes" each practice's data access** as you intended, but is in fact the way Epic's ecosystem operates in production [1] [2]. By using the Epic sandbox credentials now and planning to obtain production credentials for each client, you're on the right track for a smooth multi-tenant integration strategy.

Sources:

- Epic on FHIR Documentation – *App Registration and Client ID Distribution* [18]
- *6B Insights:* Working with Epic FHIR Endpoints – *Multi-tenant Architecture (each Epic org has separate endpoints, app registrations, and go-live process)* [5]
- Reddit (r/healthIT) – *Epic integration FAQ by industry experts (each Epic customer downloads app credentials separately via Showroom)* [3]
- *HealthPrep Codebase:* `.env` config – *Notes on Epic OAuth credentials per organization (multi-tenancy support)* [6]

---

1 5 Working with Epic FHIR Sandboxes & Production Endpoints: Best Practices - 6B
https://6b.health/insight/working-with-epic-fhir-sandboxes-production-endpoints-best-practices/

2 4 9 10 11 12 13 14 18 Documentation - Epic on FHIR
https://fhir.epic.com/Documentation?docId=testpatients

3 17 Understanding how FHIR and EPIC / Cerner and other systems work : r/healthIT
https://www.reddit.com/r/healthIT/comments/19f73ad/understanding_how_fhir_and_epic_cerner_and_other/

6 16 .env.example
https://github.com/mitfusco98/HealthPrepV2/blob/aebfee6c302220119b7ffe8f5f792852783bed3c/.env.example

7 8 fhir_client.py
https://github.com/mitfusco98/HealthPrepV2/blob/aebfee6c302220119b7ffe8f5f792852783bed3c/emr/fhir_client.py

15 Health Prep | Automated Medical Screening Preparation
https://fuscodigital.com/